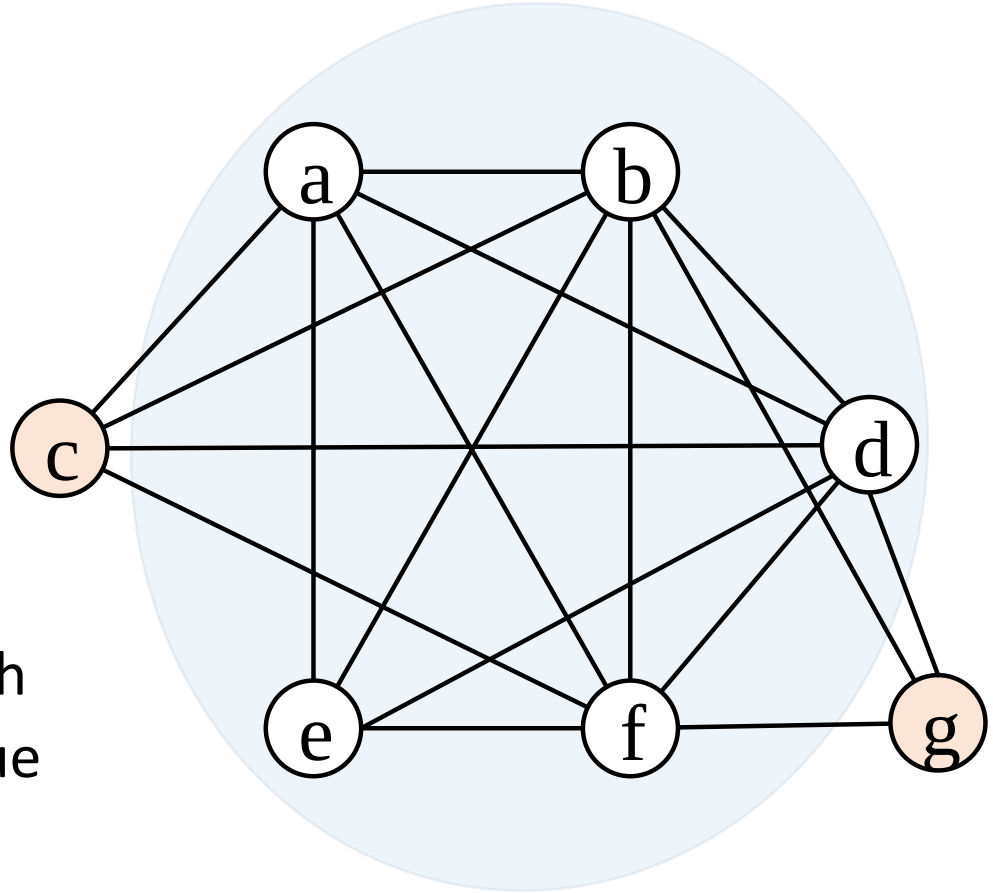
Redundancy-Aware Maximal Cliques

Jia Wang James Cheng Ada Wai-Chee Fu
Chinese University of Hong Kong

Maximal Cliques

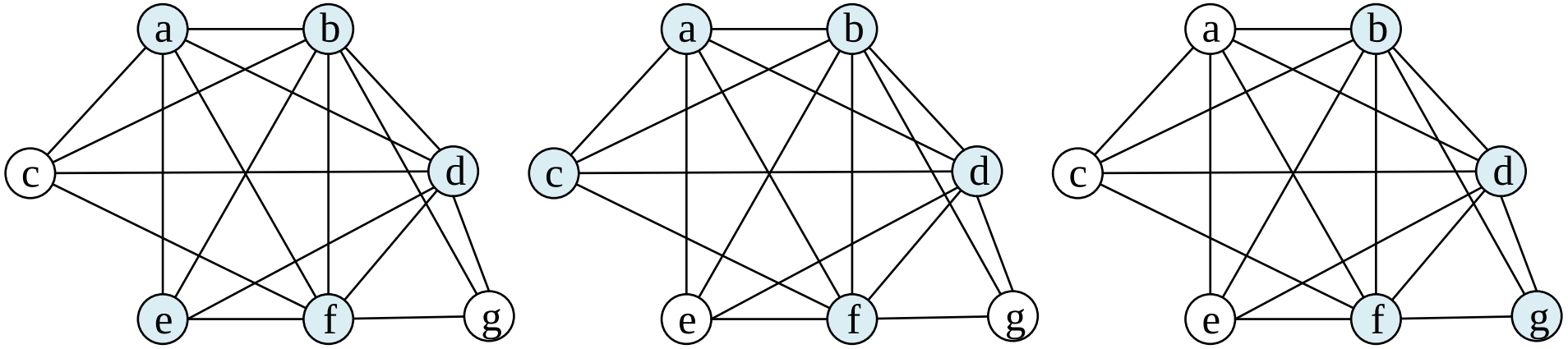
- Input
  - Undirected graph $G = (V, E)$

- Maximal cliques
  - Clique: vertex set of a complete subgraph
  - Maximal: adding vertex makes it no clique



Classic problem

- MCE (Maximal Clique Enumeration)
  - exhaustive: finding set of ALL maximal cliques

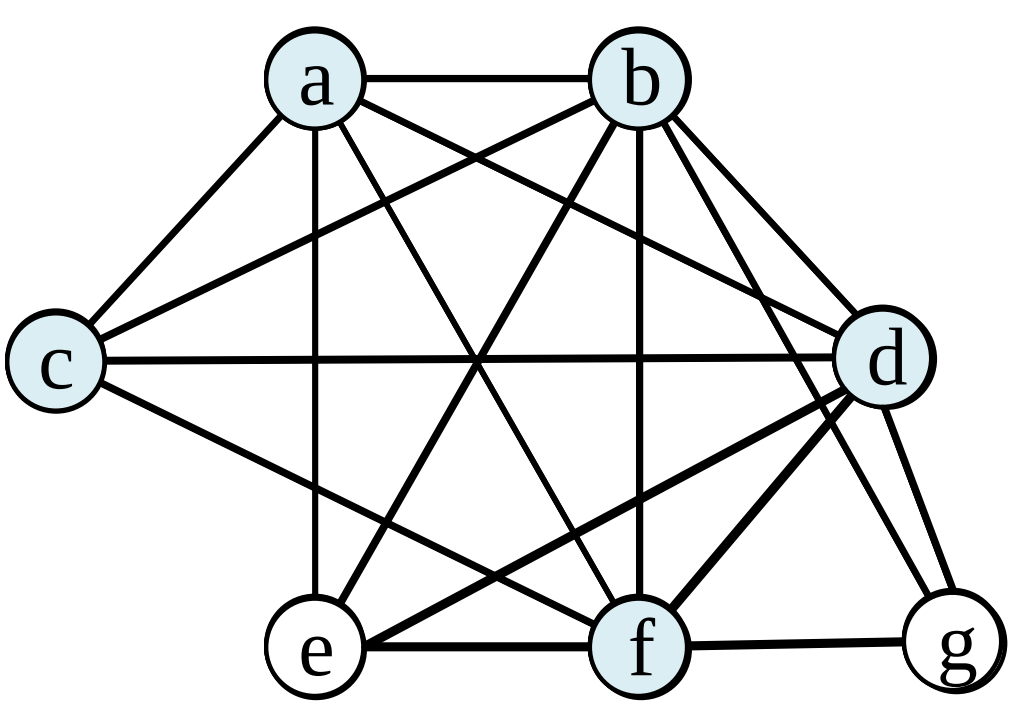


Classic algorithm

- Algorithm: recursive search
  - Maintain *current clique* C & *candidate set* T
  - Recursion:
    - select vertex in T, add to C (a branch)
    - update T

Classic algorithm

- Example

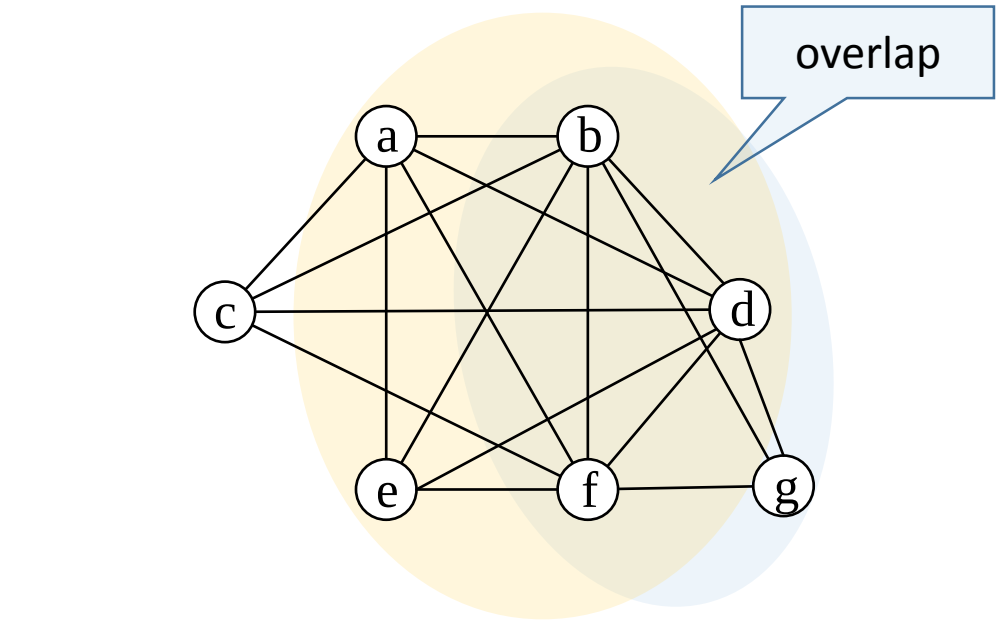


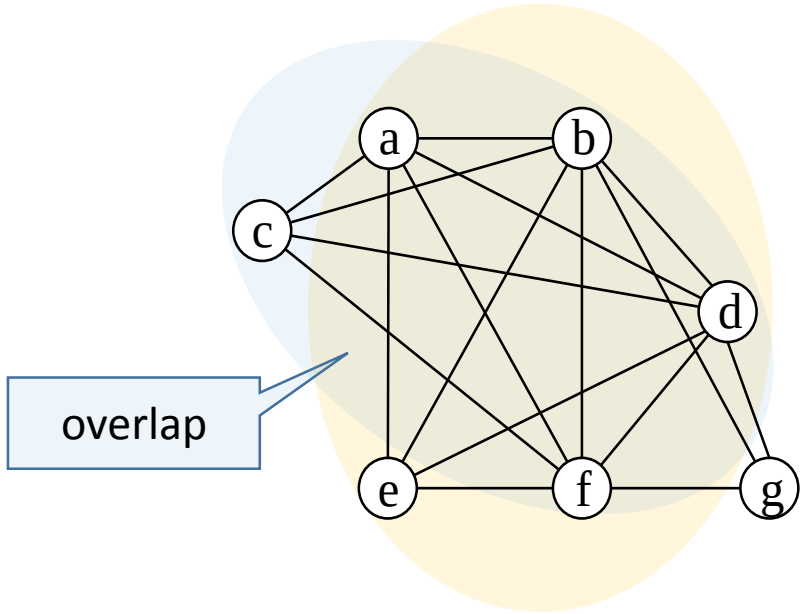
current clique

candidates

Problems of MCE

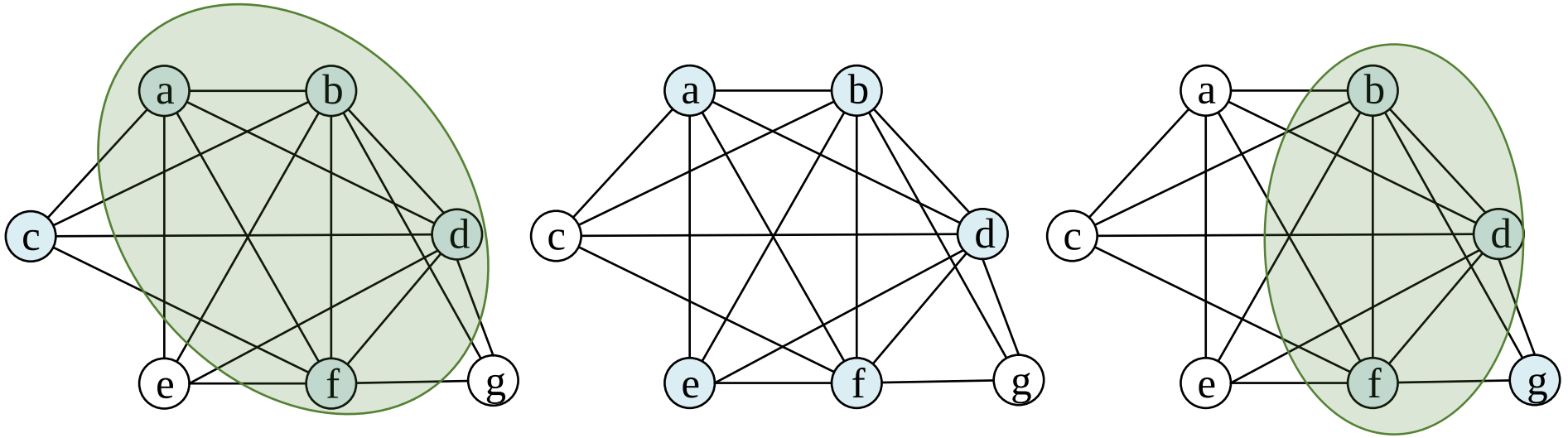
- Usability
  - overwhelmingly **large output**
  - cliques less useful due to **overlap**
  - full MCE no good or necessary
    - anomaly detection, exploration…
- Speed
  - exhaustive search of large space
  - can be *exponentially* many





Problems of MCE

- Instead we desire
  - I: compact representation – each result meaningful
  - II: preserved information – widely covering
  - I & II: a good **summary**, e.g.:

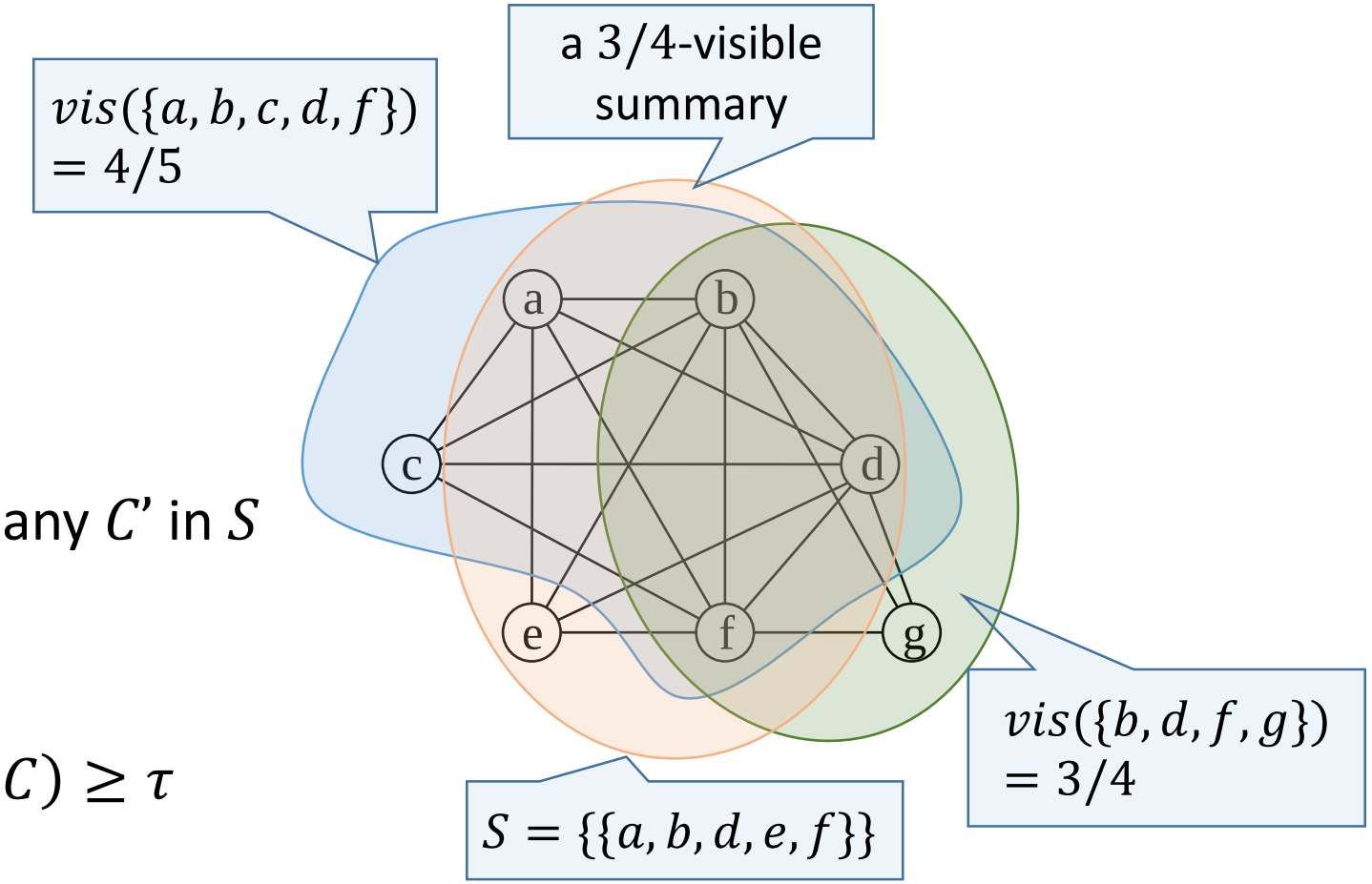


Notations

M	Set of all maximal cliques
S	a subset of M (summary)
C/C'	current/last maximal clique
r	$\frac{\lvert C' \cap C \rvert}{\lvert C \rvert}$, overlap ratio

A new notion

- *Clique* ***visibility***
  - visibility of C given S:
    max ratio r of C covered by any C' in S
  - Denoted by $vis(C)$
- ***τ-visible summary***
  - A summary S such that $vis(C) \geq \tau$
    for each C in M
- Problem: ***τ-visible MCE***
  - find a small τ-visible summary S of M



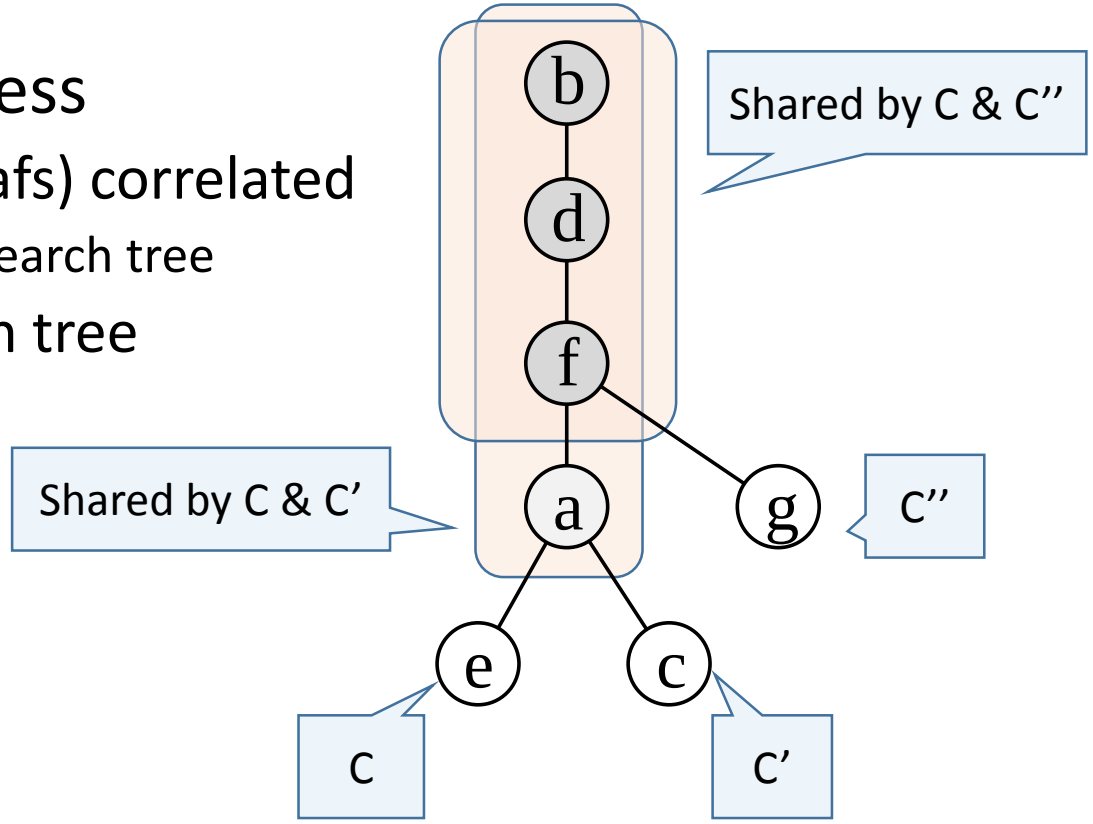
Have enabled redundancy reduction.
Possibly faster too?

A naïve implementation

- In classic MCE
  - S: summary of cliques so far
  - C compare to each maximal clique in
  - → add C to S: if no redundancy
  - → discard C: if much overlap with any C' in S
- Overhead
  - $O(T_{MCE} + |M| \times |S|)$
  - costly computation

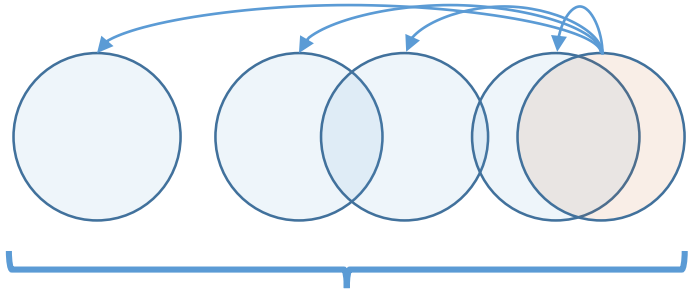
Main idea

- Characterizing search process
  - nearby cliques C and C’ (leafs) correlated
    - have *common ancestors* in search tree
  - $C \sim C$’ when close in search tree



For efficiency – first step

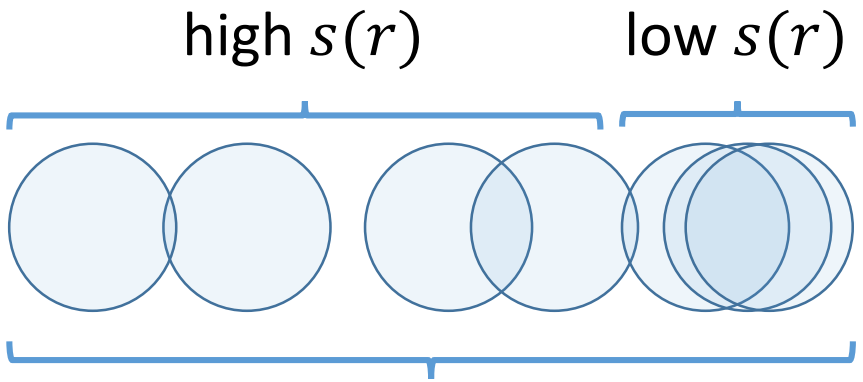
- Glancing at last one
  - discard most redundancy in one shot



generated sequence of cliques

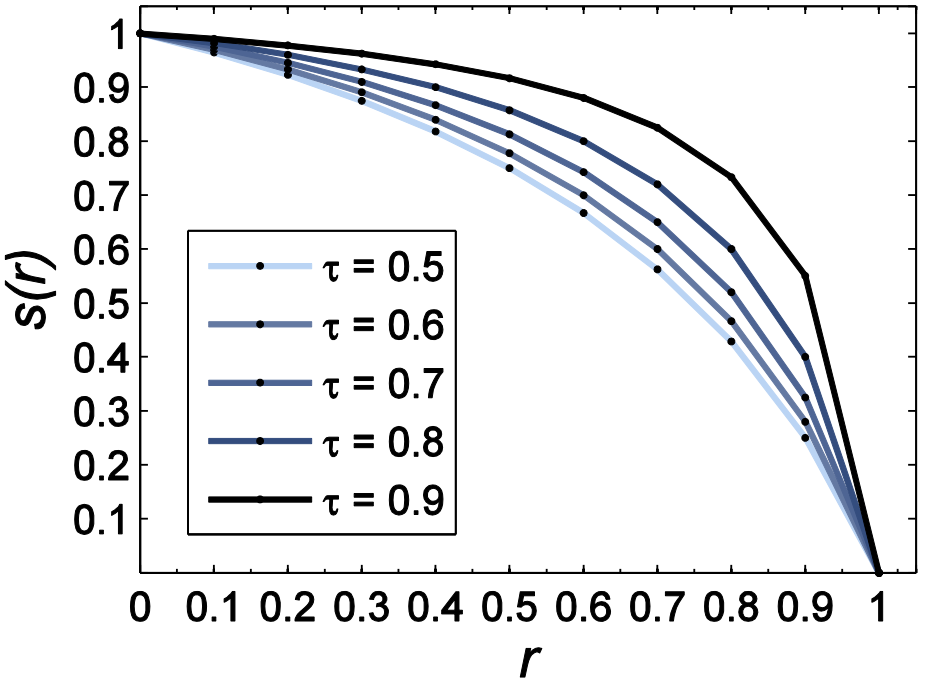
For efficiency – first step

- Summary as a sample
  - retain with probability $s(r)$: decreases with r
  - cliques as data *points*, r as *slope*
  - a perspective: analogy to *importance sampling*



For efficiency – first step

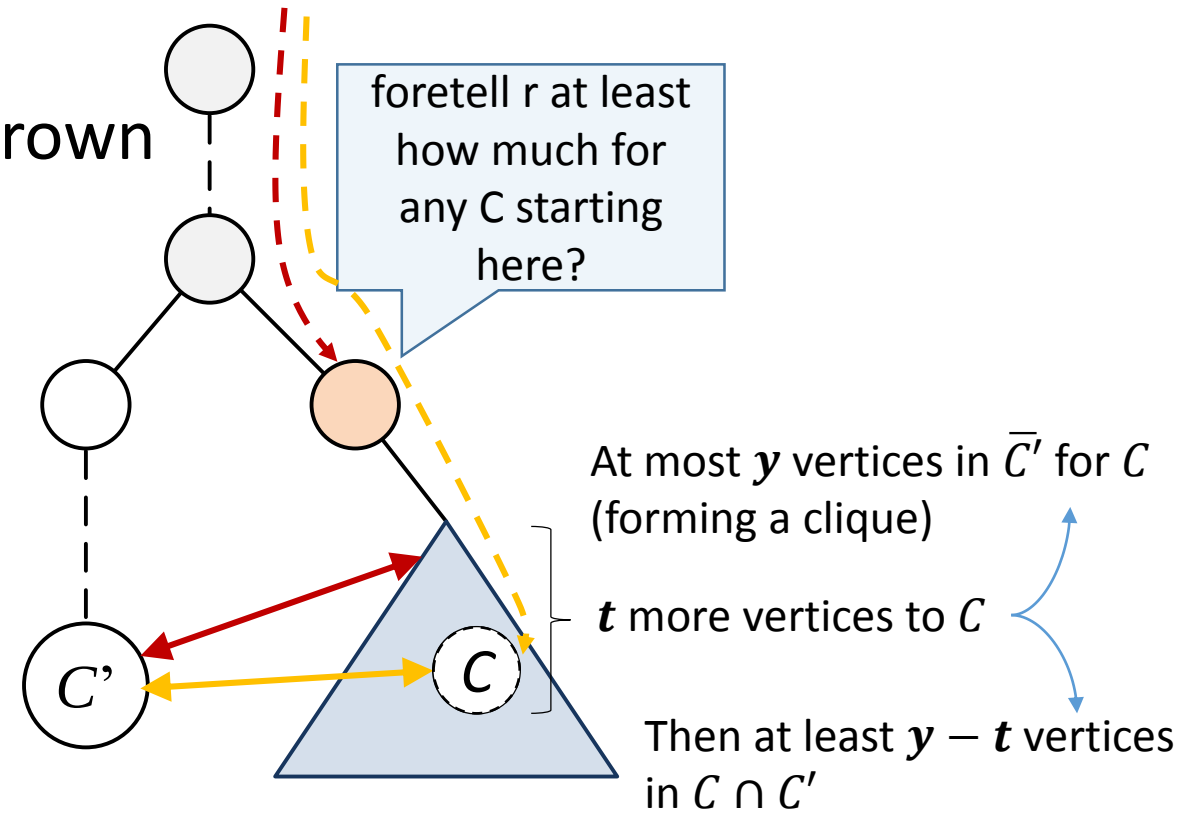
- Choice of $s(r)$
  - To meet visibility requirements
  - Choose: $s(r) = \frac{(1-r)(2-\tau)}{2-r-\tau}$
  - Claim: $E[vis(C)] \geq \tau$ for all C



For efficiency – a further step

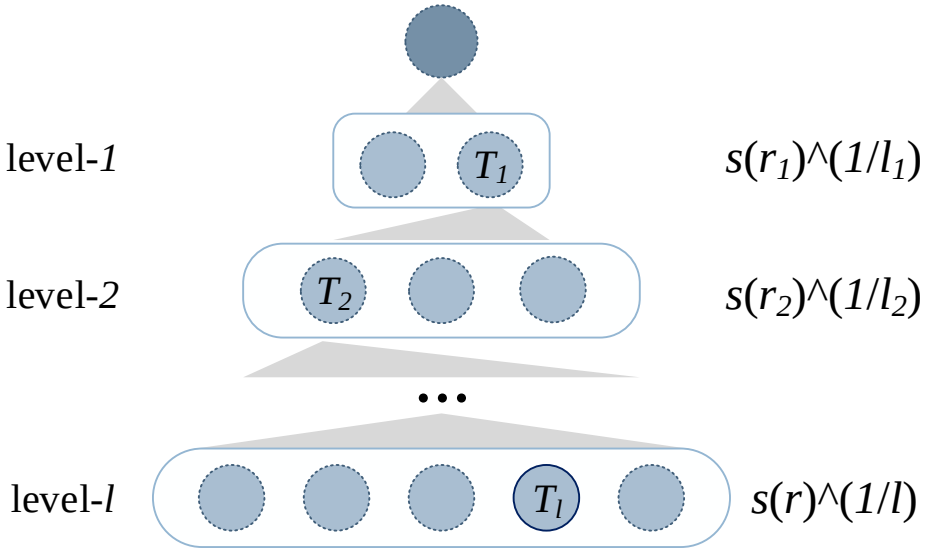
- Detected redundancy when *fully* grown
- Now: earlier with *foresight*

- At inner node
  - lower bound r
  - prune whole branch with large r



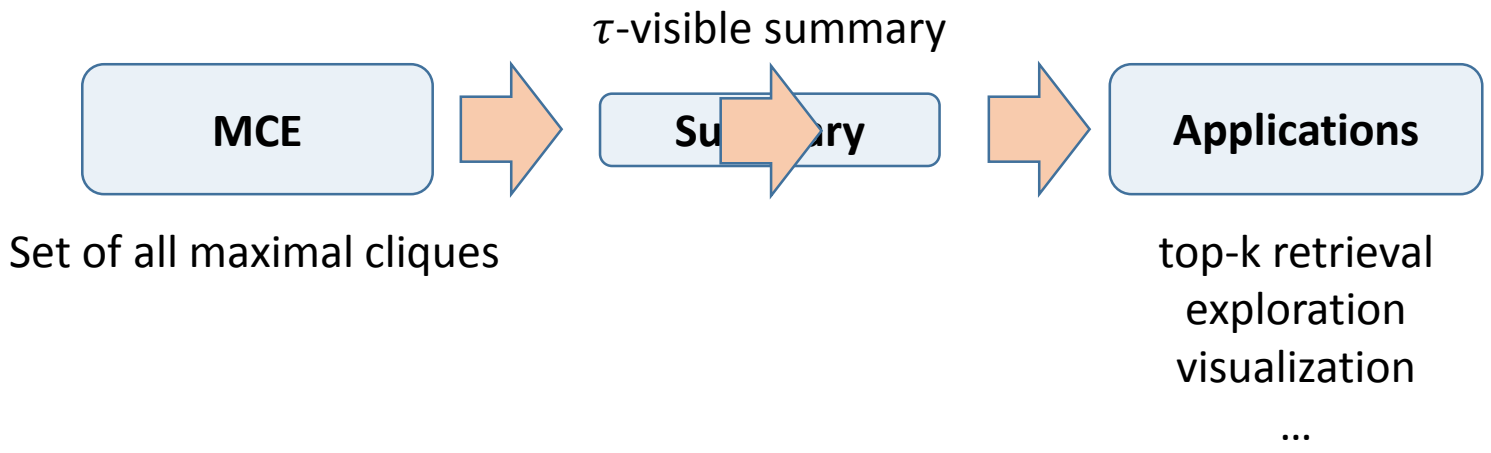
For efficiency – a further step

- Sampling search branch
  - Want: guarantee still holds
    - for expected visibility
  - Need: maintain Pr[final retaining prob.] $\geq s(r)$
  - How: set Pr[sample a branch] = $\sqrt[\bar{l}]{s(\tilde{r})}$
    - $\bar{l}$: upper bound of branch depth
    - $\tilde{r}$: lower bound of r



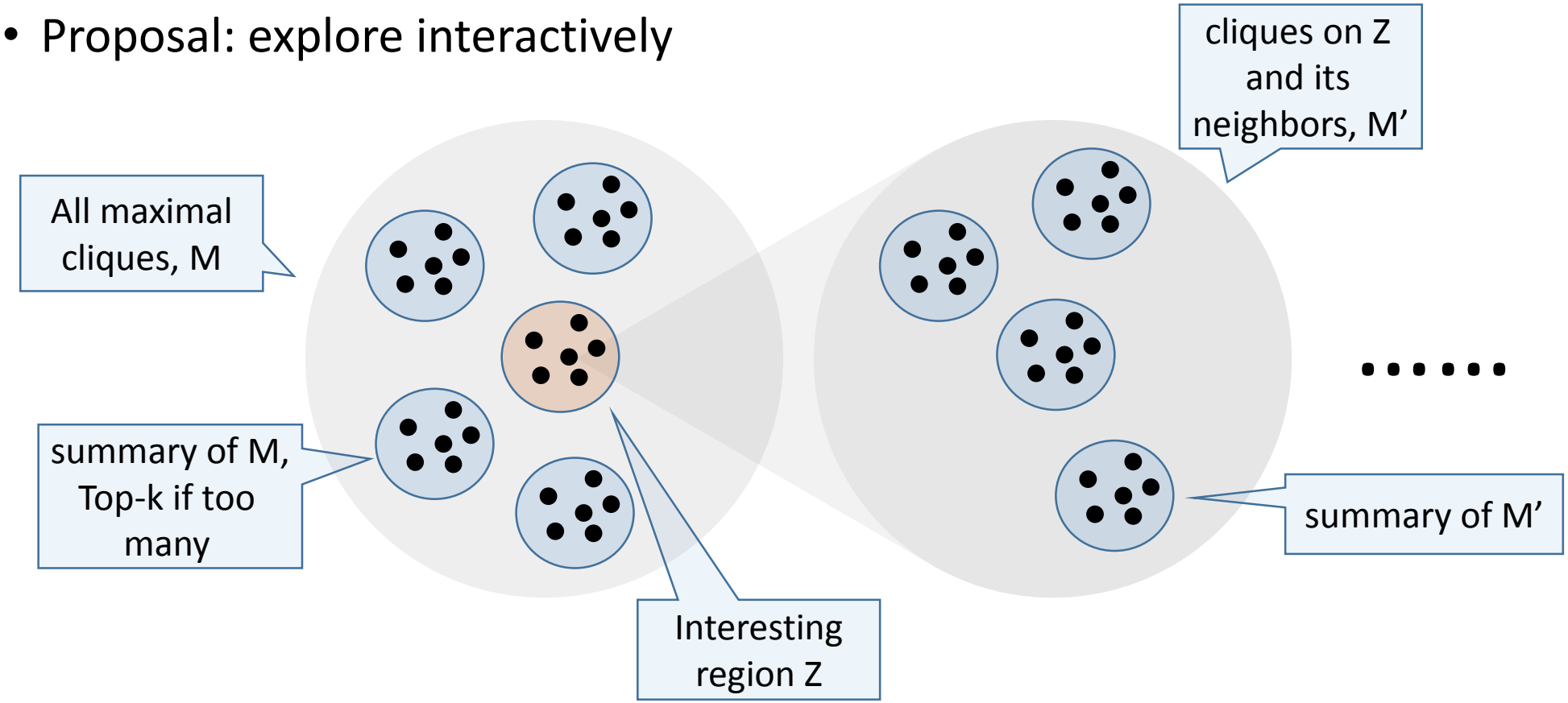
Applying the summary

- Feed other computations
  - A succinct input
  - Example: top-k results
    - Approx.` ratio using S: $\tau(1 - 1/e)$



Applying the summary

- Discovering clique space
  - Proposal: explore interactively



On real world networks

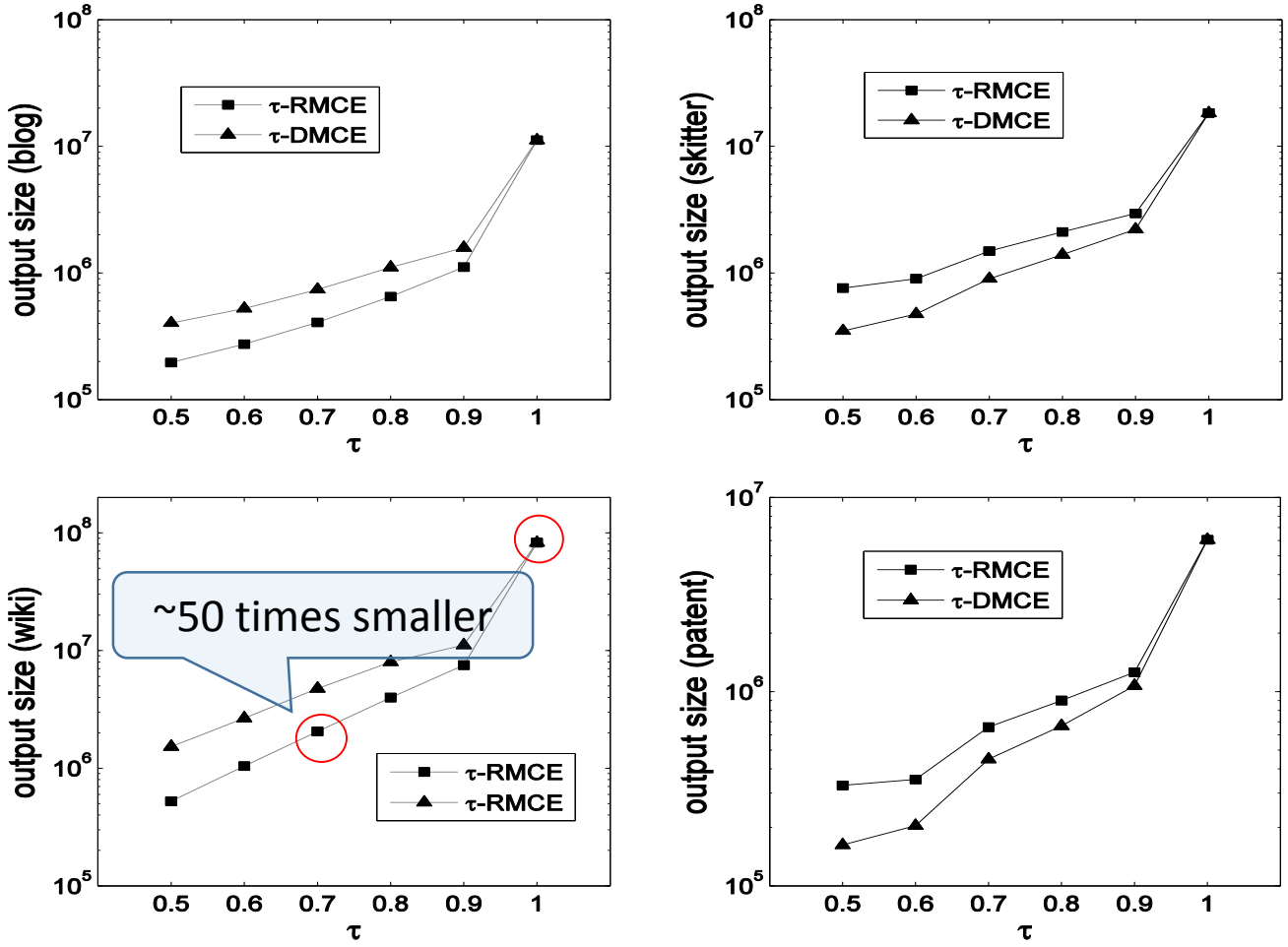
- Datasets

	Blog	Skitter	Wiki	Patent
$\|V\|$	990K	1.7M	2.4M	3.7M
$\|E\|$	6.6M	11.1M	41.7M	33M
$\|M\|$	11.2M	18.3M	82.7M	6.1M

# of all maximal cliques

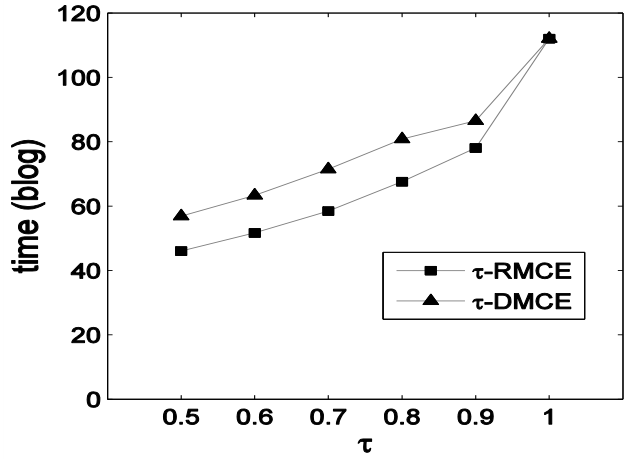
On real world networks

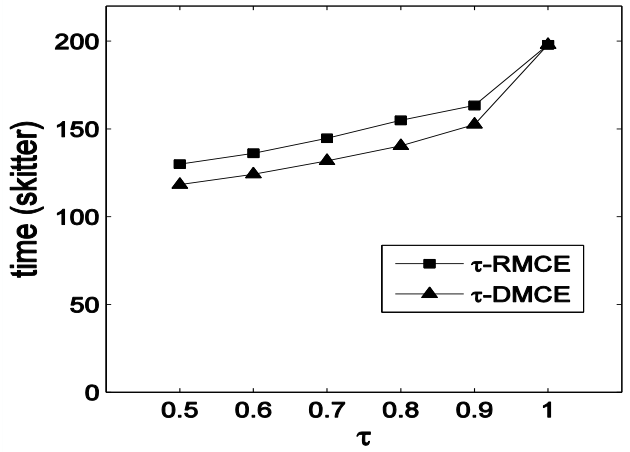
- Summary size
  - slimmed output
  - sharp drop from $\tau = 1$ to $\tau = 0.9$

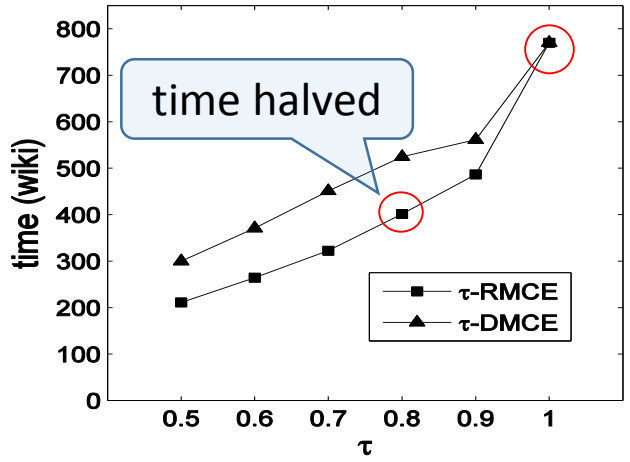


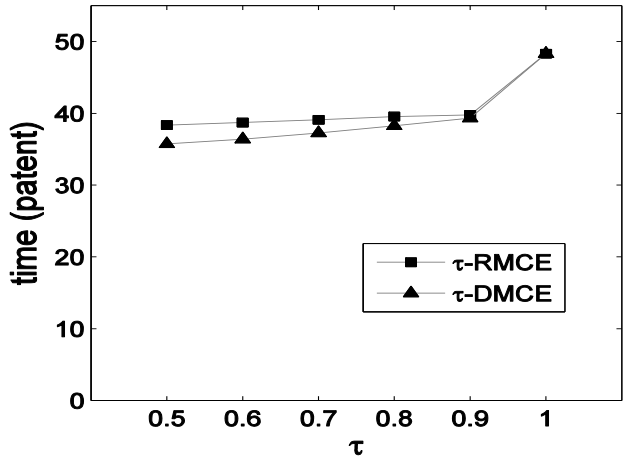
On real world networks

- Running time
  - Reduced time
  - Especially from $\tau = 1$ to $\tau = 0.9$









On real world networks

- Top-k reporting
  - using full result or summary
  - setting: $k = 20, \tau = 0.7$
  - result: small quality loss, greatly faster

	Blog	Skitter	Wiki	Patent
Q_{samp}	822	1205	462	173
Q_{all}	826	1214	464	174
T_{samp}	1.38	4.02	8.59	0.7
T_{all}	28.4	57.5	197	8.9

→ Quality by summary

→ Quality by all cliques

→ Time by summary

→ Time by all cliques

Wrapping up

- Tradeoff
  - completeness → compactness & usability & time
- Approaches
  - notion of τ-visible summary
  - fast redundancy detection
  - early pruning
  - summary as a sample
- Applications
  - exploration, top-k, and more